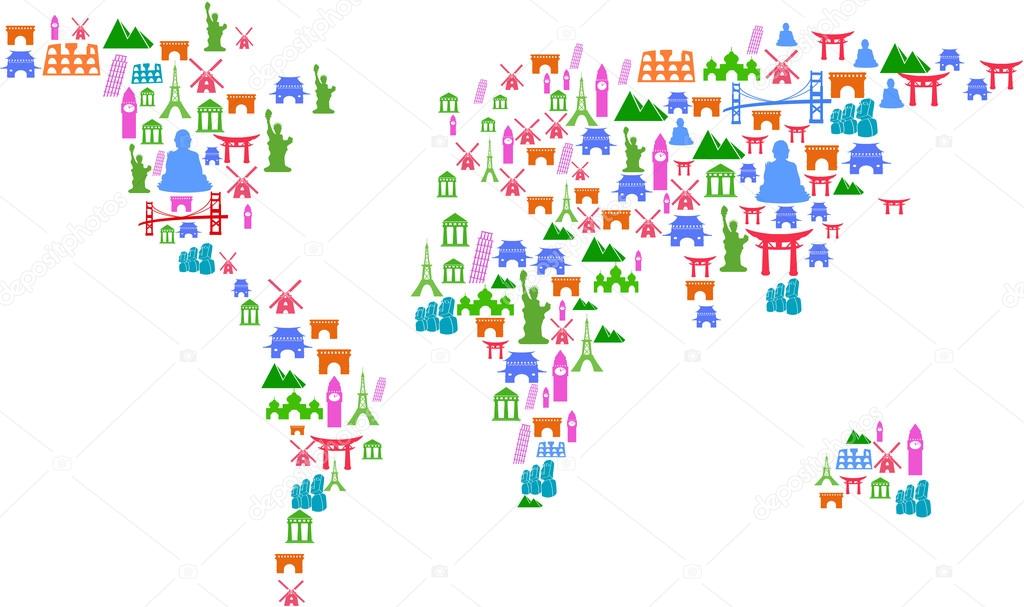
**소프트웨어 공학 팀 프로젝트**

**여행지 장소 추천 시스템**



|  |  |
| --- | --- |
| 장원희 | 심지섭 |
| 2014154035 | 2014152019 |

목차

1장. 프로젝트 개요 ------------------------------------------------------------------------------------ 3

* 1. 프로젝트 제목 1.2 선정 이유 1.3 팀 운영 방법

2장. 시스템 정의 --------------------------------------------------------------------------------------- 5

2.1 시스템 간략한 설명 2.2 유사 사례 간략한 설명

3장. 프로세스 모델 ------------------------------------------------------------------------------------ 6

3.1 규범적인 프로세스 모델 선정 및 이유 3.2 특수한 프로세스 모델 선정 및 이유

4장. 실무 가이드 원칙 -------------------------------------------------------------------------------- 8

4.1 각 프레임워크 원칙에서 중요한 3 개 정의 4.2 프로젝트 계획 보고서

5장. 요구사항 획득 ----------------------------------------------------------------------------------- 15

5.1 기능 요구사항과 비 기능 요구사항 정의

5.2 표준 양식을 사용한 시스템 요구사항 명세 3개 작성 5.3 정형적인 형식에 따른 유스케이스 작성

6장. 시스템 설계 ------------------------------------------------------------------------------------- 18

6.1 설계 개념의 중요한 개념을 적용 6.2 설계 모델에 따른 요소별 설계

7장. 아키텍처 개념 ----------------------------------------------------------------------------------- 21

7.1 아키텍처 스타일 선정 및 이유 7.2 아키텍처 설계 프로세스 정의 및 설계

중간발표 후 수정내용--------------------------------------------------------------------------------- 24

8장. 품질 ---------------------------------------------------------------------------------------------- 25

8.1 시스템 품질 속성 정의 8.2 비즈니스 품질 속성 정의 8.3 아키텍처 품질 속성 정의

8.4 소프트웨어 품질 목표, 속성과 척도 8.5 소프트웨어 통계적 방법 이용 사례 선정

9장. 테스팅 전략 ------------------------------------------------------------------------------------ 29

9.1 테스팅 전략적 이슈 순위정의 ( 전략 성공 ) 9.2 통합 테스팅 방법 및 순서

9.3 시스템 테스팅 중요이슈 정의 9.4 소프트웨어 결함 유형 정의

10장. 프로젝트 관리 --------------------------------------------------------------------------------- 32

10.1 우리 팀의 융합적인 요소[현재시점] 10.2 우리 팀의 독소 요소[시작 시점]

10.3 프로젝트 스케줄링의 활동 네트워크 작성 (태스크 일정 및 의존성 ,활동 네트워크)

11장. 시스템 유지보수 ------------------------------------------------------------------------------- 34

11.1 자동화된 유지보수 도구

12장. Review ------------------------------------------------------------------------------------------ 35

**Chapter 1**

2018.3.15 팀장 : 최한규, 서기 : 심지섭, 팀원 : 장원희

* 1. 팀 프로젝트 제목
* 여행지 장소 추천 시스템
  1. 선정 이유

|  |  |  |
| --- | --- | --- |
| 최한규 | 심지섭 | 장원희 |
| 비행기 예매 시스템 | 여행지 장소 추천 시스템 | IOT를 이용한 정기 물품구매 시스템 |
| 자동차 렌트 시스템 | 주식 투자 시스템 | 무인 편의점 관리 시스템 |
|  |  | ~~무인 물품대여 시스템~~ |

* 투표를 통해 선정하기로 결정

|  |  |
| --- | --- |
| 최한규 | 여행지 장소 추천 시스템 |
| 심지섭 | ~~무인 편의점 관리 시스템~~  여행지 장소 추천 시스템[[1]](#footnote-1) |
| 장원희 | 여행지 장소 추천 시스템 |

* 투표 결과
* 만장일치로 여행지 장소 추천 시스템 결정

1.3 팀 운영 방법

* 팀원 중 한명이 팀원역할의 사람이 이 결석한 팀원의 역할을 대신해 수행한다
* 팀원 -> 서기 -> 팀장 순으로 1주마다 순회하며 역할을 맡는다
* 해당일의 문서 작성 내용, 및 토의 내용을 카톡으로 공유한다
* *챕터 제목 : 맑은 고딕, 16pt, 진하게 (4/26 추가)*
* *글머리 양식 : 챕터 및 소제목은 1.x(■), 1.1.x(■) 식으로 작성해 감, 항목들은 검은 동그라미 기호로 통일 (5/3 추가)*
* *기본 제목 및 소제목: LG PC, 13pt 로 작성, 11pt 작성 (4/26 추가)*
* *의견 종합 표 : 중복된 의견들을 보기 쉽게 ~~주황~~빨강(5/03 수정) 으로 작성 (4/26 추가)*
* *수정 및 추가된 내용 : 본래 내용에 삭선, 이후 추가된 내용은 날짜와 함께 ~~파란색~~주황색(5/03 수정)으로 작성, 기울기 추가 (4/26 추가)*
* *항목이 많아지는 것들은 표로 작성해 보기 편하게 함 (4/26 추가)*
* *수정, 추가 및 제거된 내용들은 각주로 이유를 설명(5/3 추가)*
* *최종 결정된 내용은 ‘HY헤드라인M’(굵게)로 작성(5/3 추가)*
* *수업 내용 개인적 복습하고 서로 모르는 부분 공유, 이해하기(5/24추가)*
* *적극적 의견 표출하기, 그 의견에 대해 충분히 생각하고 의견 들려주기(5/24추가)*
* *저번 프로젝트 내용 다시 보고, 거기에 대한 의견 달아 따로 공유하기(5/24추가)*
* *다른 조들이 발표했던 내용 참고(5/24추가)*

**Chapter 2**

2018.3.15 팀장 : 최한규, 서기 : 심지섭, 팀원 : 장원희

2.1 시스템의 간략한 설명

* 블로그에 있는 여행 리뷰들을 통합운용
* 지정한 도시 근처 여행명소를 지도에 표시
* 표시된 여행 명소들을 방문하기 좋은 효율적인 여행 루트 표시

2.2 유사 사례 시스템 소개

* ~~근처 식당 위치를 알려주고 주문 대신해주는 요기요[[2]](#footnote-2)~~
* 여행지 명소에 해당하는 후기까지 서비스 해주는 TripAdvisor
* 네이버 블로그, 지식in 빅데이터 기반한 네이버 여행지 추천 서비스
* 여행지 명소를 추천해주는 투어팁스

**Chapter 3**

2018.3.15 팀장 : 심지섭 / 서기 : 장원희 / 팀원 : 최한규

3.1 규범적인 프로세스 모델 중 1개를 선정 및 이유

1. 폭포수 모델
   * 장원희 : 요구사항이 추가로 들어올 때 시스템 수정이 어려운 모델이다.
   * 심지섭 : 시작부터 모든 요구사항을 완벽히 하고 시작해야 하는데 사용자 요구가 계속 변경되므로 정하기 매우 어렵다
2. **단계적 개발 모델**
   * 심지섭 : 지도가 완성이 되야 위치 선정 가능하고, 위치 선정을 해야 루트를 만들어 서비스 해줄 수 있다. 여행에 대한 모든 기능을 통합으로 제공하는 서비스이기 때문에 사용자의 필요에 따라 부가적 기능을 추가할 필요가 있음.
   * 장원희 : 단계적 개발 모델이 사용자의 의견 수렴을 가장 잘 받아들일 수 있는 모델이라 생각.
3. V-Model
   * 장원희 : 폭포수 모델과 마찬가지로 변경되는 사용자의 요구사항을 계속 받아들이기 어렵다
4. 나선형 모델
   * 심지섭 : 지도가 계속 바뀌는 것 과 길 선정 알고리즘이 계속 업데이트 되어야 할 것 같아 검토 했지만, 지도 업데이트는 설계 변경이 아닌 DB 변경.
5. 프로토 타입 모델
   * 장원희 : 아직 개발되지 않은 분야의 시스템일 개발할 때 유용 할 것 같아 보여 우리의 시스템과는 맞지 않아 보인다.
6. Concurrent Model(동시 모델)
   * 심지섭 : 서로 종속성 이 강한 기능들이 많아 동시 동시적인 개발이 어렵다. 하지만 루트 선정 분야에서는 부분적으로 사용할 수 있을 것 같다.
   * 장원희 : 동시적으로 개발하기 힘든 시스템 같다.

* 사용자의 요구를 지속적으로 받아들일 수 있고, 요구를 받아 수정하기 쉬운 단계적 모델로 결정

3.2 특수한 프로세스 모델 중 1개를 선정 및 이유

* **Components based development model**
  + 장원희 : 지도에 위치선정을 해주는 시스템, 경로 선정 기능 같은 것들이 아무 구현된 것이 많아 초기 비용이 많이 감소될 수 있다.
  + 심지섭 : 유지보수가 용이하므로 사용자들이 원하는 기능 추가, 제거가 쉽다.

**Chapter 4**

2018.3.29 팀장 : 장원희 / 서기 : 최한규 / 팀원 : 심지섭

4.1 각 프레임워크를 가이드 하는 원칙 중 중요한 것 세가지

* 결정 방법 : 아예 겹치지 않는 항목은 건 사람마다 하나씩 결정하고, 부분적으로 남는 항목은 겹치지 않은 항목들을 대상으로 투표로 결정한다

1. 의사소통 원칙

|  |  |
| --- | --- |
| 장원희 | 심지섭 |
| **의사 결정을 메모하거나 문서화하라** | **의사결정을 메모하거나 문서화하라** |
| **초점을 유지하라. 토의를 모듈화 하라** | **초점을 유지하라. 토의를 모듈화 하라** |
| 뭔가 불분명한 점이 있으면 도식화 하라 | 의사소통 이전에 준비하라 |

* + 의사 결정을 메모하거나 문서화하라
  + 초점을 유지하라. 토의를 모듈화 하라
  + 의사소통 이전에 준비하라

1. 계획 원칙

|  |  |
| --- | --- |
| 장원희 | 심지섭 |
| 자신이 알고있는 무엇인가에 근거하여 추정하라 | 이해관계자가 계획활동에 관여하게 하라 |
| 자주 계획을 추적하고 요구에 맞게 조정하라 | 계획은 언제나 반복이라는 것을 인식하라 |
| 어떻게 품질을 보장하려는 지 정의하라 | 계획을 정의하는 것처럼 세분화를 조정하라 |

* + 장원희 : 자신이 알고있는 무엇인가에 근거하여 추정하라
  + 심지섭 : 이해관계자가 계획활동에 관여하게 하라
  + 최한규 : 계획을 정의하는 것처럼 세분화를 조정하라

1. 요구사항 원칙

|  |  |
| --- | --- |
| 장원희 | 심지섭 |
| **소프트웨어가 수행하는 기능은 반드시 정의되어야 한다** | **소프트웨어가 수행하는 기능은 반드시 정의되어야 한다** |
| 분석 작업은 필수적인 정보로부터 구현 세부 사항을 향해 전달되어야 한다. | 문제의 정보도메인은 반드시 표현되고 이해 되어야 한다 |
| 정보, 기능 및 동작을 묘하사는 모델은 단계화 된 방식에서 상세하게 드러난 방법으로 반드시 분할해야 한다. | 소프트웨어의 동작은 반드시 표현되어야 한다 |

* + 소프트웨어가 수행하는 기능은 반드시 정의되어야 한다.
  + 장원희 : 정보, 기능 및 동작을 묘하사는 모델은 단계화 된 방식에서 상세하게 드러난 방법으로 반드시 분할해야 한다
  + 심지섭 : 소프트웨어의 동작은 반드시 표현되어야 한다

1. 디자인 모델링 원칙

|  |  |
| --- | --- |
| 장원희 | 심지섭 |
| **컴포넌트 수준 설계는 기능적으로 독립되어야 한다** | **컴포넌트의 수준설계는 기능적으로 독립되어야 한다** |
| **설계 표현은 쉽게 이해될 수 있어야 한다** | **설계 표현은 쉽게 이해될 수 있어야 한다** |
| 설계는 반복적으로 개발되어야 한다 | 사용자 인터페이스 설계는 최종 사용자의 요구에 전환되어야 한다 |

* + 컴포넌트 수준 설계는 기능적으로 독립되어야 한다
  + 설계표현(모델)은 쉽게 이해할 수 있어야 한다
  + 설계는 반복적으로 개발되어야 한다

1. 구축 원칙

|  |  |
| --- | --- |
| 장원희 | 심지섭 |
| 모든 테스트는 고객 요구사항에 추적해야 한다 | 조건부 로직을 가능한한 단순하게 유지하라 |
| 완전한 테스팅은 불가능하다 | 설계 요구사항을 충족할 데이터구조를 선택하라 |
| 결함을 추적하고 테스팅을 통해 발견된 결함에서 패턴을 찾아라 | 코드를 재구축하라 |

* + 장원희 : 결함을 추적하고 테스팅을 통해 발견된 결함에서 패턴을 찾아라
  + 심지섭 : 조건부 로직은 가능한한 단순하게 유지하라(겹침), 코드를 재구축하라
  + 최한규 : 조건부 로직은 가능한한 단순하게 유지하라

1. 배치 원칙

|  |  |
| --- | --- |
| 장원희 | 심지섭 |
| **소프트웨어에 대한 고객의 기대는 반드시 관리되어야 한다** | **소프트웨어에 대한 고객의 기대는 반드시 관리되어야 한다** |
| 완성된 배포 패키지는 조합되고 테스트되어야 한다 | 지원체제는 반드시 소프트웨어가 배포되기 전에 확립되어야 한다 |
| 적절한 교육자료는 반드시 최종 사용자에게 제공되어야 한다 | 결함이 있는 소프트웨어는 먼저 수정된 뒤 배포되어야 한다 |

* + 소프트웨어에 대한 고객의 기대는 반드시 관리되어야 한다
  + 지원체제는 반드시 소프트웨어가 배포되기 전에 확립되어야 한다
  + 완성된 배포 패키지는 조합되고 테스트되어야 한다.

4.2 프로젝트 계획 보고서 작성

1. 개요
   1. 목표 : 여행지 장소 추천 시스템 프로토타입 완성
   2. 제약조건

|  |  |
| --- | --- |
| 예산 | * 급여 : 200만원 \* 3명 \* 3개월 * AWS 서버 대여비용 * 디자인 외주비용 |
| 시간 | 3개월 |
| 인원 | 3명 |

1. 프로젝트 조직
   1. 관리

|  |  |
| --- | --- |
| 역할 | 프로젝트 전체적인 관리 |
| 담당 인원 | 심지섭 |

* 1. 개발

|  |  |
| --- | --- |
| 역할 | 요구사항 식별, 분석, 설계, 프로그래밍 |
| 담당 인원 | 장원희, 최한규, 심지섭 |

* 1. 외주 관리 및 서버 대여

|  |  |
| --- | --- |
| 역할 | 개발에 필요한 AWS 서버, 디자인 외주를 관리 |
| 담당 인원 | 최한규 |

* 1. 테스트

|  |  |
| --- | --- |
| 역할 | 개발된 각 모듈 및 완성된 제품을 테스트 |
| 담당 인원 | 장원희 |

1. 위험 분석

|  |  |
| --- | --- |
| 팀원이 도중 드랍 하였을 때 | * + - 포기하려는 팀원을 설득해본다. 그래도 포기하려 하면 팀원의 역할을 조금이라도 덜어 다른 팀원이 가진다.     - 그래도 포기할 시 기존 조직 역할 중 팀원을 없애고 팀장, 서기 조직으로만 운영한다 |
| 의견 충돌로 인한 팀 내 분쟁 | * + - 두 명이 분쟁 시 나머지 한명이 분쟁을 없애기 위해 노력한다.     - 모든 팀원끼리 불화가 생길 시 교수님께 상담한다. |

1. 하드웨어 소프트웨어 자원 요구사항
   1. 개발 할 수 있는 PC 3대
   2. AWS 서버(DB)
   3. 안드로이드 스튜디오
2. 업무 분할

|  |  |
| --- | --- |
| 개발 시작일 ~ 15일 | * + - 요구사항 분석     - 설계     - 디자인 외주 |
| 15일 ~ ~~80~~[[3]](#footnote-3)*55*일 | * + - 코딩     - 완성된 기능마다 테스트     - 코딩, 테스트 반복 |
| ~~80일 ~ 90일[[4]](#footnote-4)~~  *55일 ~ 65일* | * + - 외주 받은 디자인 적용     - 최종 테스트 |

1. 프로젝트 일정
   1. 요구사항 분석

|  |  |
| --- | --- |
| 예상 시간 | 10일 |
| 담당 인원 | 심지섭 |

* 1. 설계

|  |  |
| --- | --- |
| 예상 시간 | 5일 |
| 담당 인원 | 심지섭 |

* 1. 디자인 외주

|  |  |
| --- | --- |
| 예상 시간 | 1일 |
| 담당 인원 | 심지섭, 최한규 |

* 1. 코딩

|  |  |
| --- | --- |
| 예상 시간 | 10일 |
| 담당 인원 | 장원희, 최한규, 심지섭 |

* 1. 완성된 기능 테스트

|  |  |
| --- | --- |
| 예상 시간 | 3일 |
| 담당 인원 | 장원희 |

* 1. 외주 받은 디자인 적용

|  |  |
| --- | --- |
| 예상 시간 | 3일 |
| 담당 인원 | 장원희, 최한규 |

* 1. 최종 테스트

|  |  |
| --- | --- |
| 예상 시간 | 7일 |
| 담당 인원 | 장원희, 최한규, 심지섭 |

1. 감시와 보고 체계
   * 각 단계의 진행중, 완성된 제품들을 구글 드라이브의 기능으로 공유
   * 수정사항이 생겼을 때 공유 문서에 수정했다는 사실을 알 수 있게 수정(삭선, 수정한 날짜)
   * 테스트 한 기능의 버그들은 카카오톡 방에 전송해 모두가 즉시 알 수 있게 함

**Chapter 5**

2018.4.5 팀장 : 최한규 / 서기 : 심지섭 / 조원 : 장원희

5.1 기능 요구사항과 비기능 요구사항 정의

1. 기능 요구사항
   * 블로그에 있는 여행 리뷰들을 통합운용
   * 지정한 도시 내 명소를 지도에 표시
   * 명소를 선택하면 명소의 리뷰를 나열
   * 표시된 여행 명소들을 방문하기 좋은 효율적인 여행 루트 표시
2. 비기능 요구사항

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 제품 요구사항 | |  |  | | --- | --- | | 성능 요구사항 | * + - * 원하는 도시를 선택하면 1초안에 명소가 지도에 표시된다       * 표시된 명소를 클릭하면 2초안에 관련된 리뷰가 나열된다       * 사용자가 원하는 명소를 지정하면 4초 안에 경로를 표시한다 | | 이식성 요구사항 | * + - * 안드로이드 사용자의 99%에 해당하는 젤리빈(4.1버전) 이상부터 지원한다 | |
| 조직 요구사항 | |  |  | | --- | --- | | 배포 요구사항 | * + - * 기본적으로 안드로이드 마켓을 이용       * Lite 버전과 advanced 버전으로 나누어 배포 | | 구현 요구사항 | * + - * 안드로이드 스튜디오 사용       * Kotlin 언어 사용 | |
| 외부 요구사항 | |  |  | | --- | --- | | 성능 요구사항 | * + - * 다른 사이트 리뷰를 표절하는 사용자는 이용불가 조치 한다       * 리뷰로 악의적으로 명소에 안 좋은 평을 주는 것을 막는다 | |

5.2 표준양식을 사용한 요구사항 명세서 작성

<장원희>

|  |  |
| --- | --- |
| 기능 | 명소의 리뷰를 나열한다. |
| 설명 | 사용자가 명소를 클릭하면 추천순으로 리뷰를 나열한다. |
| 입력 | 사용자가 클릭한 명소 |
| 출처 | 클라이언트에서 온 패킷  DB쿼리 결과 |
| 출력 | 추천순으로 정렬된 리뷰의 리스트 |
| 목적지 | 사용자의 화면 |
| 액션 | 1. 사용자가 명소를 선택한다. 2. DB에서 해당 명소에 대한 리뷰를 가져온다   리뷰들이 10개가 넘어가면, 10개를 리뷰를 하나의 페이지로 선정한다. |
| 필요사항 | 리뷰들이 많을 시 사용자가 읽기 편하게 페이지 단위로 표시한다 |
| 사전조건 | 명소의 리뷰가 존재해야 한다  사용자가 도시를 선택해 놓은 상태여야 한다 |
| 사후조건 | 사용자의 화면이 리뷰의 리스트가 된다. |
| 부작용 | 없음 |

<심지섭>

|  |  |
| --- | --- |
| 기능 | 지도에 선택한 장소 주변의 명소출력 |
| 설명 | 선택한 도시의 랜드마크와 명소 Top10출력 |
| 입력 | 검색장소의 위치(추가조건:검색범위,테마,) |
| 출처 | 장소리뷰 DB, 지도의 기본 랜드마크 데이터 |
| 출력 | 지도 결과창에 Dot으로표시 – 장소의 위치 |
| 목적지 | 검색 결과창(앱의 베이스가 되는 지도창) |
| 액션 | 1. 일반적으로 사용되는 랜드마크의 위치를 출력한다. 2. 리뷰수와 평점수를 종합하여 Top10을 출력한다. 테마가 정해진경우 검색범위를 줄이고, 검색범위안의 장소들만 종합한다. 리뷰가 많은장소를 채도가 높은 빨간색으로 표시한다. |
| 필요사항 | 입력한 값을 기반으로 DB의 정보를 결과창 맵에 출력할 수 있도록 한다. |
| 사전조건 | 검색하는 장소,랜드마크가 존재하는 곳이어야 하고, Top10출력은 리뷰10개, 별점3개 이상이어야 한다. |
| 사후조건 | 기본 랜드마크만 있던 맵에 리뷰위치가 Dot으로 출력된다. |
| 부작용 | 지도창의 이동 시 이동속도가 저하된다. |

5.3 정형적인 형식에 따른 유스케이스 작성

<장원희>

|  |  |
| --- | --- |
| 제목 | 명소의 리뷰를 출력 |
| 개요 | 사용자가 리뷰를 원하는 명소를 클릭한다 |
| 선행조건 | 명소에 등록된 리뷰가 존재해야 한다 |
| 관련 액터 | 앱 사용자, DB |
| 작업 흐름 | 1. 사용자가 표시된 명소중 원하는 것을 클릭한다 2. 시스템은 DB쿼리를 통해 명소에 해당하는 리뷰들을 찾는다 3. 시스템이 화면에 리뷰들을 추천순으로 나열한다 4. 만약 리뷰가 10개 이상이라면, 10개의 리뷰를 한 페이지로 놓고 페이지 단위로 표시해준다 |
| 기타 요구사항 | 사용자 요청부터 출력까지의 시간은 1초 이내여야 한다 |

<심지섭>

|  |  |
| --- | --- |
| 제목 | 장소 Top10출력 |
| 개요 | 장소 중 검색조건에 맞는 장소Top10을 자동선정해 출력한다 |
| 선행조건 | DB에 접근할 수 있고, 조건에 맞는 장소 중 적어도 하나가 리뷰10개, 별점3개이상이어야 한다 |
| 관련액터 | DBMS, 검색 창 |
| 작업흐름 | 1. 검색창에서 받은 장소와 제약조건을 DB에 전송한다. 2. DBMS는 해당 장소와 입력받은 범위(Default:1km) 내에 있는 장소들을 검색한다. 3. 리뷰중 출력조건에 맞는 Top10항목을 우선적으로 찾아놓는다. 4. 출력값을 지도창에 출력한다. |
| 기타 요구사항 | 사용자와 앱, 앱과 DB의 연결이 원활해야한다. |

**Chapter 6**

2018.4.12 팀장 : 심지섭 / 서기 : 장원희 / 조원 : 최한규

6.1 설계 개념의 중요한 개념을 적용

1. 선정 방법 : 다수결

|  |  |  |
| --- | --- | --- |
| 장원희 | 심지섭 | 최한규 |
| **관점** | **관점** | **관점** |
| **기능적 독립성** | **기능적 독립성** | **기능적 독립성** |
| **모듈화** | **모듈화** | **모듈화** |
| **추상화** | **추상화** | 정제 |
| **정보 은닉** | 아키텍쳐 | **정보 은닉** |
| 패턴 | **관점분리** | **관점분리** |

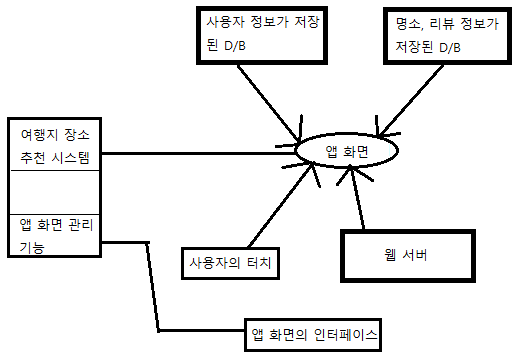
* **관점 / 모듈화 / 기능적 독립성 / 추상화 / 정보은닉 / 관점분리**

6.2 설계 모델에 따른 요소별 설계

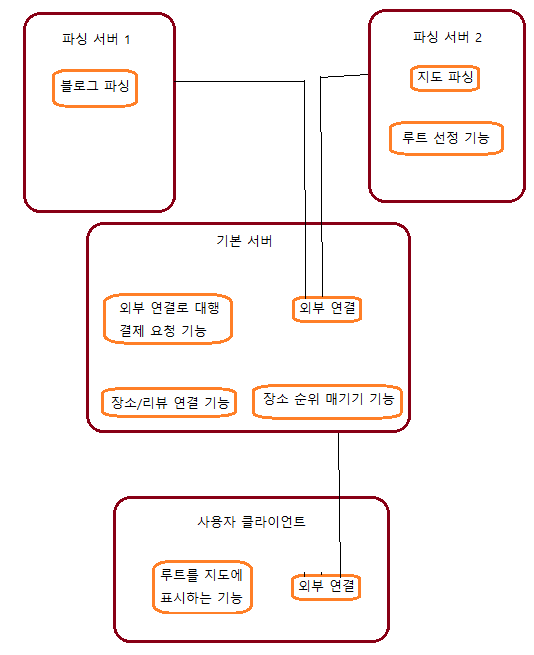
1. 데이터 설계 요소

|  |  |
| --- | --- |
| 객체 | 루트 |
| 속성 | 운행 수단 |
| 교통비 |
| 출발지 |
| 경유지 |
| 명소 |
| 걸리는 시간 |

1. 인터페이스 설계 요소



1. 컴포넌트 수준 설계 요소
   * 블로그 리뷰 파싱 기능
   * 가져온 블로그 리뷰를 명소별로 나누는 기능
   * 리뷰 관심도 순 정렬 기능
   * 불법 광고 차단 기능
   * 루트 선정 기능
   * 루트 표시 기능
   * 지도 파싱 기능
   * 장소에 대한 리뷰를 연결
   * 대행 결제 기능
   * 명소 순위 매기는 기능
2. 배치 수준 설계 요소



**Chapter 7**

4/19 팀장 : 장원희 / 서기 : 최한규 / 조원 : 심지섭

7.1 아키텍처 스타일 선정 및 이유

1. 데이터 중심 아키텍처

|  |  |
| --- | --- |
| 장원희 | 심지섭 |
| 기능들간 공유해야 하는 데이터가 많지 않아 데이터 저장소의 필요성이 떨어진다 | 우리 시스템에서 데이터도 중요하지만 데이터를 사용자의 요구사항에 맞춰서 변화시키는게 주 기능이기때문에 그렇게 유용한 시스템은 아니다 |

1. **데이터 흐름 아키텍처**

|  |  |
| --- | --- |
| 장원희 | 심지섭 |
| 우리 S/W 는 각 컴포넌트들의 출력 값이 다른 컴포넌트의 입력 값으로 들어가는 상황이 대부분이다. | 특정형태의 데이터를 입력받고, 특정형태를 만족하는 데이터를 필터를 이용해 만들어내는 것에 특화되어 있다. |

1. 호출 및 복귀 아키텍처

|  |  |
| --- | --- |
| 장원희 | 심지섭 |
| 데이터 흐름과 비교했을 때 기능 추가 제거가 비교적 쉽지만, 우리 S/W의 아키텍처가 데이터 흐름 아키텍처에 더 맞는 것 같다. | 서로 다른 기능의 컴포넌트들의 종합적인 결과를 하나로 통합하여 출력하는 우리 시스템에는 적합하지 않은 방법이다. |

1. 객체지향 아키텍처

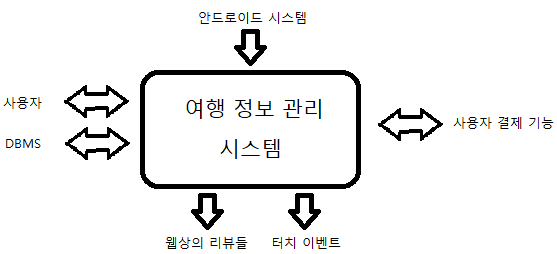
|  |  |
| --- | --- |
| 장원희 | 심지섭 |
| 컴포넌트들을 객체화 시키는데 요구되는 초기 비용(시간)이 크다. 빠른 개발과 서비스에 적합하지 않다. | 컴포넌트 사이의 교류와 협력을 메시지를 활용하는데 컴포넌트가 다루는 정보의 양이 너무 많아 전달하고 수신하는데 오랜 시간이 소요된다. |

1. 층위구조 아키텍처

|  |  |
| --- | --- |
| 장원희 | 심지섭 |
| 우리 S/W는 컴포넌트들을 계층화 해야 할 정도로 다양한 기능들을 포함하지 않는다. | 입력이 들어오면 그 즉시 여러개의 컴포넌트들로 분산되어 일을 수행하는 우리 시스템에서 층을 여러 개 만드는 것 보다는 각 컴포넌트들의 역할과 통합이 중시되어야 한다. |

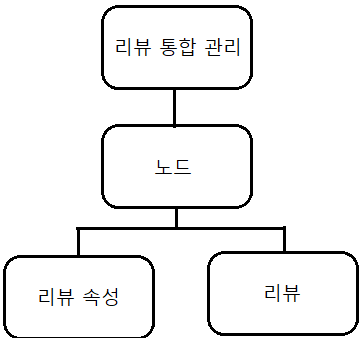
7.2 아키텍처 설계 프로세스 정의 및 설계

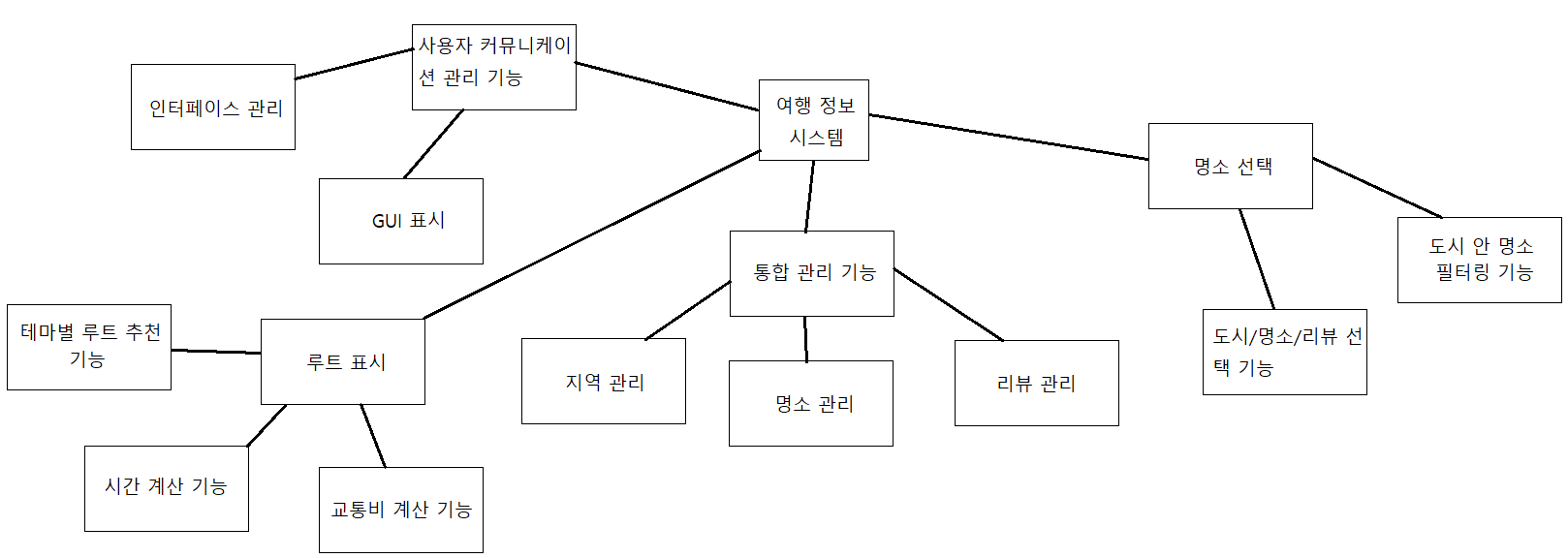
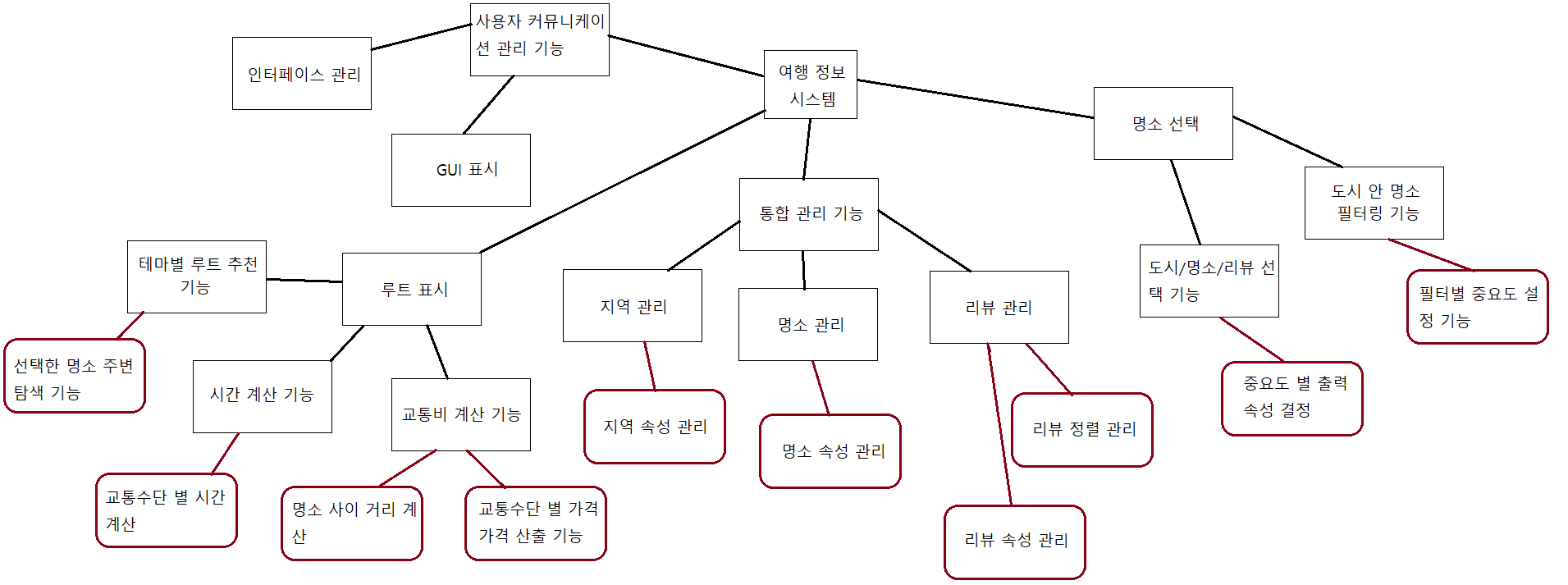
1. 시스템 배경도



1. 원형 정의
   1. 여행 정보 관리 시스템의 원형 : 리뷰를 통합 관리

|  |  |
| --- | --- |
| 리뷰통합 관리 | 모든 관리 요소들을 통합적으로 관리 |
| 노드 | 리뷰와 리뷰 속성을 합쳐 관리를 도와줌 |
| 리뷰 | 각 리뷰를 구성하는 요소들 |
| 리뷰 속성 | 평가 점수, 포함된 명소, 리뷰 같은 속성 |



1. 아키텍처를 컴포넌트로 정제
2. 시스템 실현

**중간 발표 후 수정 내용**

1. 의견

|  |  |
| --- | --- |
| 장원희 | 심지섭 |
| 문서 작성 양식을 표준화한 것 팀 운영방법에 추가 | 표 및 다른 자료들을 활용하고, 각 자료들의 위치선정으로 인해 보기에 편함 |
| 각 문서들의 페이지 하단에 페이지 수 표시 | 문서에 컬러가 있어서 보는데 지루하지 않음 |
| 변경되거나, 추가된 요소들을 글씨 및 기울기로 표현한 것 |  |

2. 토의

|  |  |
| --- | --- |
| 고려 내용 | 결과 |
| 문서 작성 양식을 표준화한 것 팀 운영방법에 추가 | 문서 작성 내용도 |
| 각 문서들의 페이지 하단에 페이지 수 표시 | 페이지 수 넣지 않아도 알아보기 불편하지 않음 |
| 변경되거나, 추가된 요소들을 글씨 및 기울기로 표현한 것 | 수정되거나, 추가된 것을 한눈에 알아볼 수 있음 |
| 표 및 다른 자료들을 활용하고, 각 자료들의 위치선정으로 인해 보기에 편함 | 우리 문서의 알아보기 보기 어려운 양식을 표를 이용해 보기 편하게 작성 |
| 문서에 컬러가 있어서 보는데 지루하지 않음 | 컬러를 활용해 문서의 가독성을 높인다 |

3. 결정

* 챕터 제목 : 맑은 고딕, 16pt, 진하게
* 의견 종합 표 : 중복된 의견들을 보기 쉽게 주황색으로 작성
* 글머리 양식 : 챕터 및 소제목은 1.x, 1.1.x 식으로 작성해 감, 항목들은 검은 동그라미 기호로 통일
* 수정 및 추가된 내용 : 본래 내용에 삭선, 이후 추가된 내용은 날짜와 함께 파란색으로 작성, 기울기 추가
* 기본 제목 및 소제목: LG PC, 13pt 로 작성
* 항목이 많아지는 것들은 표로 작성해 보기 편하게 함

**Chapter 8**

1. 시스템 품질 속성 정의
   1. 의견

|  |  |
| --- | --- |
| 장원희 | 심지섭 |
| **성능** | **성능** |
| 사용 편의성 | 변경 용이성 |

* 1. 결정

|  |  |
| --- | --- |
| 고려 내용 | 이유 |
| 성능 | 여행가서 시간에 쫓기는 상황이 많아서 성능 중요 |
| 사용 편의성 | 사용 빈도가 높지 않아 직관적이지 않으면 사용자가 사용하기 불편 |
| 변경 용의성 | 계속해서 기능이 추가되어야 하기 때문 |

1. 비즈니스 품질 속성 정의
   1. 의견

|  |  |
| --- | --- |
| 장원희 | 심지섭 |
| **시간 적시성** | **시간 적시성** |
| 신규 발매 일정 | 목표 시장 |

* 1. 결정

|  |  |
| --- | --- |
| 고려 내용 | 이유 |
| 시장 적시성 | 완성 단계를 졸업시즌으로 맞추면서 광고해서 더 많은 사람들이 관심 갖게 한다. |
| 신규 발매 일정 | 신규 발매 일정을 여행에 관심이 쏠리는 시즌으로 맞춘다. |
| 목표 시장 | 여행을 자주가는 계층 조사해 그 쪽 중심으로 시장 선정 |

1. 아키텍처 품질 속성 정의
   1. 의견

|  |  |
| --- | --- |
| 장원희 | 심지섭 |
| **정확성과 안정성** | **정확성과 안정성** |
| **개발 용이성** | **개발 용이성** |

* 1. 결정

|  |  |
| --- | --- |
| 고려 내용 | 이유 |
| 정확성과 안정성 | 여행에 관한 정보가 잘못되면 사용자에 피해가 생기고 그에 따라 신뢰성이 낮아짐 |
| 개발 용이성 | 짧은 기간의 프로젝트 개발 기간에 완성하는게 중요 |

1. 소프트웨어 품질 목표, 속성과 척도
   1. 의견

|  |  |  |
| --- | --- | --- |
|  | 장원희 | 심지섭 |
| 요구사항 품질 | **즉흥도** | **즉흥도** |
| 추적도 | 완성도 |
| 설계 품질 | **구조 보전도** | **구조 보전도** |
| **연결 완성도** | **연결 완성도** |
| 코드 품질 | **재사용성** | **재사용성** |
| 유지보수 | 완성도 |
| QC 효율성 | **효율적 테스트** | **효율적 테스트** |
| 완성율 | 자원 할당 |

* 1. 결정

|  |  |  |
| --- | --- | --- |
| 요구사항 품질 | 즉흥도 | 사용자 요구사항에 중점을 두고 있기 때문에 |
| 추적도 | 요구사항을 중점에 두고 있기 때문에 요구사항과 설계/코드가 다르면 안된다. |
| 완성도 | 연결해야 할 컴포넌트가 많기 때문에 협의나 결정할 것이 많음 |
| 설계 품질 | 구조 보전도 | 모델이 있어야 개발할 때 효율적으로, 품질을 높여 개발할 수 있기 때문이다. |
| 연결 완성도 | 많은 컴포넌트들로 이루어져 있어 연결 완성도가 중요하다 |
| 코드 품질 | 재사용성 | 컴포넌트를 많이 가질수록 개발 기간이 단축된다 |
| 유지보수 | 버그가 줄어 테스트 기간이 짧아지고 개발 기간이 단축된다 |
| 완성도 | 요구사항이 자주 추가되고 그거에 대한 유지보수가 이루어지기 편해야 한다. |
| QC 효율성 | 효율적 테스트 | 예산이 부족하고 시간이 촉박할 것이기 때문에, 예상을 벗어나면 개발이 힘들어진다 |
| 완성율 | 메인 기능이 명확하므로 거기에 많은 자원 할당이 필요하다 |
| 자원 할당 | 예상시간 내에 개발하기 위해 테스트가 효율적으로 이루어 져야 함 |

1. 통계적 방법 이용 사례 선정
   * 의견

|  |  |
| --- | --- |
| 장원희 | 심지섭 |
| 컴포넌트 접속의 비 일관성 | **안 정하거나 잘못된 명세** |
| 프로그래밍 표준 위반 | **고객 대화의 잘못된 이해** |
| **고객 대화의 잘못된 이해** | 데이터 재표현의 오류 |
| **안 정하거나 잘못된 명세** | 논리설계의 오류 |
| **인간/컴퓨터 간의 일관성 없음이나 모호함** | **인간/컴퓨터 간의 일관성 없음이나 모호함** |

* + 결정

|  |  |
| --- | --- |
| 고려 내용 | 이유 |
| 인간/컴퓨터 간의 일관성 없음이나 모호함 | 다른 것보다 사용하는 사람들의 편의가 우선시 되어야 하는 소프트웨어이기 때문 |
| 안 정하거나 잘못된 명세 | 명세가 잘못되었다면 필연적으로 구축 및 테스팅 과정에서 많은 오류가 생겨 시간 일정에 차질이 많이 생김 |
| 고객 대화의 잘못된 이해 | 요구사항이 엇나가면 이미 많은 대체제가 있는 시장에서 도태될 것 |
| 컴포넌트 접속의 비 일관성 | 많은 컴포넌트들로 구축될 우리 소프트웨어 특성상 중요함 |
| 데이터 재표현의 오류 | 하나의 데이터를 계속해 가공해 사용하는데 거기서 불필요 데이터가 나올 수 있음 |

**Chapter 9**

1. 테스팅 전략적 이슈 순위 정의
   * 의견

|  |  |
| --- | --- |
| 장원희 | 심지섭 |
| **자기 자신을 테스트하도록 설계된 견고한 소프트웨어 개발** | **자기 자신을 테스트하도록 설계된 견고한 소프트웨어 개발** |
| **소프트웨어의 사용자를 이해하고 각 사용자 분류 별 프로파일을 개발할 것** | **소프트웨어의 사용자를 이해하고 각 사용자 분류 별 프로파일을 개발할 것** |
| 테스트 프로세스를 지속적으로 개선하는 방안을 개발할 것 | 테스팅의 목적을 명확하게 서술할 것 |

* + 결정

|  |  |
| --- | --- |
| 고려 내용 | 이유 |
| 자기 자신을 테스트하도록 설계된 견고한 소프트웨어 개발 | 테스트 과정을 넣음으로써 오류에 대한 빠른 대처가 가능하다.  결과 나오기 전의 중간 값을 체크해 오류를 줄일 수 있다. |
| 소프트웨어의 사용자를 이해하고 각 사용자 분류 별 프로파일을 개발할 것 | 여행이란 주제가 너무 광범위 하기 때문에 프로파일을 만들어 분류할 필요가 있다. |
| 테스트 프로세스를 지속적으로 개선하는 방안을 개발할 것 | SW가 개선될수록 발생 할 수 있는 오류의 수가 많아지므로 그에 대한 테스트 프로세스도 계속 개선 |
| 테스팅의 목적을 명확하게 서술할 것 | 코더와 테스터가 다를 때 테스트의 목적을 다르게 둘 수 있다 |

1. 통합 테스팅 방법 및 순서
   * 의견

|  |  |
| --- | --- |
| 장원희 | 심지섭 |
| **스모크 테스팅** | **스모크 테스팅** |

* + 결정

|  |  |
| --- | --- |
| 결정 사항 | 이유 |
| 스모크 테스팅 | 잦은 테스트는 빠른 피드백을 가능하게 한다. 빠듯한 개발 시간 때문에 우리 시스템이 개발 단계에서 걸러내지 못한 오류가 있을 수 있고, 그것을 보완 할 수 있다.  상황이 진전되고있다는 좋은 지표를 제공 함으로서 개발완료까지 팀의 분위기를 좋게 유지할 수 있다. |

* + 테스팅 프로세스
    1. 코드로 변환된 소프트웨어 컴포넌트를 빌드로 통합
    2. 일련의 테스트는 빌드의 기능을 제대로 수행하지 못하게 하는 에러들을 찾아내기 위해 설계
    3. 빌드는 다른 빌드와 통합되고, 전체 제품은 매일 스모크 테스팅

1. 시스템 테스팅 중요 이슈 정의
   * 의견

|  |  |
| --- | --- |
| 장원희 | 심지섭 |
| **성능 테스팅** | **성능 테스팅** |
| 복구 테스팅 | 배치 테스팅 |

* + 결정

|  |  |
| --- | --- |
| 고려 내용 | 이유 |
| 성능 테스팅 | 사용자가 여행 갔을 때 급박한 상황에 처할 확률이 높아 빠른 성능을 요함, 요구했던 성능 요구사항과 많이 벗어나면 안됨 |
| 복구 테스팅 | 급박한 상황에서의 빠른 복구가 이루어지지 않으면 사용자 불만이 큼 |
| 배치 테스팅 | 실제 사용자가 어떤 방식으로 사용하는지에 대한 정보가 필요 |

1. 결함 유형 정의
   * 의견

|  |  |
| --- | --- |
| 장원희 | 심지섭 |
| **알고리즘 결함** | **알고리즘 결함** |
| 문서화 결함 | 스트레스 또는 과부하 결함 |
| 하드웨어 및 시스템 소프트웨어 결함 | 계산 및 정밀도 결함 |

* + 결정

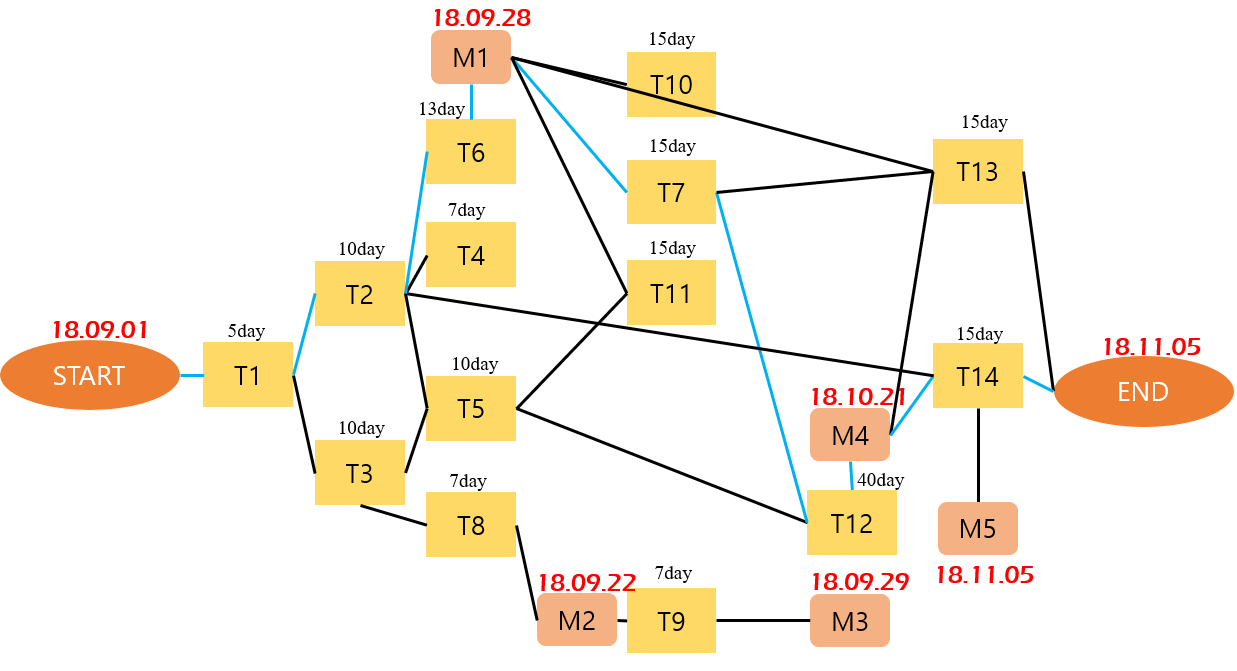
|  |  |
| --- | --- |
| 고려 내용 | 이유 |
| 알고리즘 결함 | 결함에 의해 사용자가 핵심 기능인 최적 루트 찾기에 계속 피해 받을 경우 우리 제품을 쓸 이유가 없음 |
| 계산 및 정밀도 결함 | 맵 경로 및 탐색에서 길에 대한 명확하지 않은 정보로 사용자에게 잘못된 정보 전달 가능성 있다 |
| 하드웨어 및 시스템 소프트웨어 결함 | 안드로이드 개발 특성상 잦은 버전 변경으로 기존 코드가 제대로 동작하지 않을 수 있음 |
| 문서화 결함 | 문서화 자체가 잘못되었을 경우 그 이후의 모든 개발이 크게 어긋날 수 있다. 개발 속도가 느려 질 것이다. |

**Chapter 10**

1. 우리 팀의 융합적인 요소
   * 기존 정의된 역할들의 역할을 명확히 정의
   * 수업 내용 개인적 복습하고 서로 모르는 부분 공유, 이해하기
   * 적극적 의견 표출하기, 그 의견에 대해 충분히 생각하고 의견 들려주기
   * 저번 프로젝트 내용 다시 보고, 거기에 대한 의견 달아 따로 공유하기
   * 다른 조들이 발표했던 내용 참고
2. 우리 팀의 독소 요소
   * 불분명하게 정의된 소프트웨어 팀 내에서의 역할
   * 목요일 회의 전 사전 회의 부족
   * 경직된 분위기
   * 프로젝트에 대한 이해 부족
   * 문서 작성 능력 부족
3. 프로젝트 스케줄링의 활동 네트워크 작성
   1. 태스크 일정 및 의존성

|  |  |  |  |
| --- | --- | --- | --- |
| Activity | | Duration(Day) | Dependencies |
| T1 | 프로젝트 개요 정의 | 5일 |  |
| T2 | 시스템 정의 | 10일 | T1 |
| T3 | 프로세스 모델 정의 | 10일 | T1 |
| T4 | 실무 가이드 원칙 정의 | 7일 | T2 |
| T5 | 프로젝트 계획 보고서 | 10일 | T2, T3 |
| T6 | 기능/비기능 요구사항 정의 | 13일 | T2(M1) |
| T7 | 시스템 설계 | 15일 | T6 |
| T8 | 아키텍처 스타일 선정 | 7일 | T3 |
| T9 | 아키텍처 설계 프로세스 정의 및 설계 | 7일 | T8(M2) |
| T10 | 품질 정의 | 15일 | T6 |
| T11 | 테스팅 전략 정의 | 15일 | T5, T6(M3) |
| T12 | 코딩 | 40일 | T5, T7(M4) |
| T13 | 테스트 케이스 설계 | 15일 | T6, T7, T12 |
| T14 | 통합 테스팅 | 15일 | T2, T12(M5) |

* 1. 활동 네트워크



**Chapter 11**

1. 자동화된 유지보수 도구
2. 의견

|  |  |  |
| --- | --- | --- |
| 장원희 | 파일 비교기 | 자주 변경되는 우리 소프트웨어의 특성상 |
| **교차 참조 발생기** | 점진적 개발로 계속해 컴포넌트가 추가. 각 컴포넌트의 설계, 미치는 영향을 아는 것 중요 |
| 정적 코드 분석기 | 잦은 변경과 테스팅 필요. 경로의 수 같은 구조적 정보를 알면 테스팅 하기 수월하다 |
| 심지섭 | 디버깅 도구 | 어플리케이션을 조작하기 때문에, 실행환경과 제작 환경이 달라서 적합한 디버깅 도구가 필요 |
| **교차 참조 발생기** | 변경 수행이 일어났을 때의 연쇄적인 데이터 변동이 많아서 일일이 찾는 작업이 쉽지 않다. |
| 형상관리 저장소 | 우리 시스템이 기능이 다양하지 않고, 하나의 메인 기능위주로 동작하기 때문에 형상관리 DB가 쌓이게 되면 문제를 찾고 수정하는데 큰 도움이 된다. |

1. 결정

|  |  |
| --- | --- |
| 결정 항목 | 이유 |
| 교차 참조 발생기 | 개발 프로세스 특성상 변경, 수정이 잦다. 교차 참조 발생기는 변경, 수정에 걸리는 시간을 단축시켜 준다. |
| 디버깅 도구 | 어플리케이션을 조작하기 때문에, 실행환경과 제작 환경이 달라서 적합한 디버깅 도구가 필요 |
| 정적 코드 분석기 | 잦은 변경과 테스팅 필요. 경로의 수 같은 구조적 정보를 알면 테스팅 하기 수월하다 |
| 형상관리 저장소 | 우리 시스템이 기능이 다양하지 않고, 하나의 메인 기능위주로 동작하기 때문에 형상관리 DB가 쌓이게 되면 문제를 찾고 수정하는데 큰 도움이 된다. |

Review 2014154035 장원희

2학년까지 80학점 정도의 수업 과목들을 들으면서 많은 팀 프로젝트가 있었습니다. 하지만 올해 소프트웨어 공학에서 했던 팀 프로젝트는 특이하다고 느꼈습니다. 지금까지의 팀 프로젝트는 팀원과의 소통 과정은 상관없이 결과물만 좋으면 좋은 점수를 받을 수 있었고, 만약 내용에 따라오지 못하는 팀원이 있다면 나머지 팀원들끼리 열심히 해서 좋은 프로젝트를 만드는 것이 가능 했고, 실제로 제가 했던 팀 프로젝트 중 절반 정도는 그렇게 해결 했습니다. 하지만 이번 팀 프로젝트는 달랐습니다. 교수님이 한학기 내내 말씀하셨던 인성, 협동성이 무엇보다 중요했습니다. 팀 프로젝트의 평가 기준이 얼마나 서로 의견을 많이 나누었는가, 참여는 적극적으로 했는지가 중요 평가의 척도였기 때문에 기존 팀 프로젝트를 하던 식으로 하면 결코 좋은 점수를 받을 수가 없었습니다. 하지만 저는 이번 프로젝트에서도 옛날에 했던 방식을 고집했었고 중간고사 시험을 볼 때 아차 싶었습니다. 팀원간 대화는 뒤로하고 좋은 보고서 작성에만 중점을 두고, 조원 중 한명이 아예 참여하지 않기 시작한 뒤부터는 의욕도 떨어져 전 수업시간 공부도 따로 하지 않고 목요일 회의에서 별 진전 없이 한시간을 보내고 간 적도 많았습니다. 그렇게 중간고사를 망치고 나니 보고서에 대한 생각이 달라졌습니다. 그 이후부터는 어떤 항목을 결정해도 저의 의견과 팀원의 의견을 대조하고, 서로가 옳다, 틀리다 생각하는 부분에 대해 의견을 나누어 결과물을 결정하니 지금까지 해왔던 보고서 작성과 중간고사 이후에 작성하는 보고서는 겉으로 보기에는 변경된 템플릿으로 인한 가독성 향상 이외에는 달라진 점이 없었지만 저와 팀원이 목요일 회의 이후 느끼는 점은 확실히 달라졌습니다. 교수님이 중간 발표 때 말씀하셨던 “내용이 중요한 것이 아니라 어떻게 토의했는지가 중요한 것이다” 라는 말도 이해가 되었습니다. 서로 의견을 나누니 저번 수업에 했던 내용 복습도 잘되고, 다른 사람은 이 항목에 대해 이렇게 생각할 수도 있다고 생각도 되었습니다. 솔직히 팀원 중 한명이 아예 포기하고 두 명이서 보고서를 작성할 때 솔직히 의견 모으기 귀찮았는데 차라리 잘됐다는 생각도 했었지만 막상 적극적으로 의견을 나누기 시작하니 그 사람이라면 여기서 어떤 의견을 제시했을까 하고 생각이 들며 너무나도 아쉬웠습니다.

물론 위에서 말한 것 이외에도 이 프로젝트에서 얻은 것이 많다고 생각합니다. 다른 사람이 보기 좋고 이해하기 쉽게 문서를 작성하는 법을 조금이나마 더 알게 되었습니다. 특히 중간 발표 전과 발표 후의 저희 보고서를 대조해보면, 같은 내용이여도 표로 정리하고, 미리 정의된 템플릿대로 문서를 작성해 나가면서 확실히 보기 쉽게 달라졌습니다.

만약 제가 다음에도 비슷한 프로젝트를 하게 된다면 일단 첫 단추를 잘 끼기 위해 아는 사이는 아니더라도 같이 수업을 들으며 성적 잘 받고 열심히 했던 사람들을 기억해 내서 조원부터 잘 구성하고 싶습니다. 또한 위에서 느낀 것처럼 이제는 보기 좋은 보고서만 작성하는 것이 아니라 그 보고서가 나오는 과정을 중요시 하며 프로젝트를 완성해 나가고 싶습니다.

Review 2014152019 심지섭

수업의 3분의1을 할애하면서까지 매번 회의를 진행하는 이유는 무엇일까. 나는 그것이 궁금해졌다. 소프트웨어공학. 이 수업은 어떻게 하면 소프트웨어를 잘, 효율적이게 만들 수 있는지를 가르쳐주는 수업이다. 맨 처음 교수님께서 작은 개집을 만들 때에는 계획같은 것을 열심히 세울 필요가 없지만, 큰 건물을 지을 때는 계획과 설계가 필수적이라고 말씀하셨다. 소프트웨어를 만드는 것도 마찬가지다. 규모가 큰 프로젝트에서는 사람도 많아지고, 이에 따라 적절한 계획과 분배가 있어야 사람들에게 일을 효율적으로 시킬 수 있다. 즉 많은 사람과 일을 하는 것이 필수가 될 것이기 때문에 회의를 한다는 것, 자신의 의사를 표출하고, 다른 사람의 의견과 조율하고 소프트웨어를 만들기 위한 시뮬레이션과정을 거친다는 것이 중요하고 필요한 과정이었다. 그래서 팀 프로젝트로 매주 한 시간씩 회의를 진행했던 것이다. 회의에서 가장 중요한 것은 일단 팀원이다. 우리 조는 원래 세명이었지만 한명이 거의 안 나오다시피 해서 두 명이서 프로젝트를 진행하게 되었다. 서로 처음 보았기 때문에 최대한 피해를 주기 싫었지만 내가 실력이 많이 떨어졌기 때문에 정말 열심히 공부했다. 수업내용은 복습을 못해도 회의준비는 꼬박꼬박 해갔다. 상대방도 매번 열심히 준비해오는게 느껴졌고, 그래서 회의분위기는 좋았다. 정말 찰떡궁합인 사람도 있기 마련이지만 매번 회의상대가 그럴 수는 없는 노릇이다. 서로 상대방을 배려하고, 회의를 열심히 준비해가는 것 그것이 첫번째가 되어야한다고 느꼈다. 회의는 나를 족쇄처럼 감싸 바른 생활을 하게 만들었다. 목요일 1교시는 정말 지각하고 싶은 시간이지만 조원이 두 명이기 때문에 내가 지각하면 회의를 진행할 수가 없다. 그래서 지각하지 않도록 할 수밖에 없었고, 중간고사에도 팀 회의에 관한 질문들이 있어서, 중간고사 공부도 할 수 밖에 없었다. 내가 점수 낮게 받는 건 상관이 없지만 나 때문에 상대방이 점수가 낮게 나오는 것은 너무 미안했기 때문이다. 더욱이 상대방은 공부를 잘하는 사람이었기 때문에 내가 점수를 깎아 먹으면 안됐었다. 챕터10의 프로젝트 관리에 대한 회의를 진행할 때는 자기반성의 모습을 띄었다. 우리 팀의 독소요소를 정하는데 자기 팀의 문제를 잡아내지 않으면 안되었었기 때문이다. 일단 내가 용기내서 말했다. 지각해서 회의가 늦게 시작하는 것 같다고. 그랬더니 상대방도 자신의 부족한점을 털어놓는 것이 아닌가. 꽤 민감한 주제였음에도 불구하고 우리 팀의 회의분위기는 부드러웠다. 매번 자신의 부족한점을 남에게 지적 받는 것에 익숙해져 있던 나의 모습을 조금이나마 극복하는 기회가 되었다. 교수님께서는 회의에서 나온 결과보다 수업에서 배운 것을 다시 보고, 사용해 보는 것이 중요하다고 말씀 하셨지만, 나에게는 회의라는 것 자체에 대한 모습이나 생각이 더 크게 다가왔다. 다음학기부터 팀을 이루어 졸업작품을 수행해야 되는 부담감 때문일까, 회의를 시작하기전 사전준비는 어떻게 해가야 되는지, 팀원의 역할분담을 어떻게 나눠야 지만 한사람이 지치지 않고 모두 같은 무게를 가질 수 있는지, 팀원 각각의 지식의 차이가 얼마나 나는지에 따라 회의의 양상이 어떻게 변화하는지 같은 회의에 관한 생각들을 정리하면서 여럿이서 협업하는 것을 간접적으로 나마 경험할 수 있었다. 발표를 할 때에 ‘설민석’처럼 말하는 발표자를 보았다. 별로 좋은 인상은 받지 못했던 것 같지만 자기조의 발표내용에 대해 얼마나 애정이 있고 내용을 얼마나 잘 전달하고 싶었으면 저렇게 할 수 있을까 라는 생각이 들었다. 그리고 같은 수업을 들었음에도 불구하고 내용뿐만 아니라, 문서를 만드는 양식, 회의의 진행방식까지 다른 것을 보고, 앞으로 회의를 진행하는데 있어 ‘정도’란 없고 구성원들에 따라서 얼마든지 바뀔 수 있겠구나 라는 생각을 가지게 되었다. 하지만 회의로 전 수업내용을 복습하기에는 요구사항이 수업 전체적인 내용을 담고있지 않아 좀 맞지 않았던 것 같다.

1. 다른 조원들의 의견을 듣고 결정 변경 [↑](#footnote-ref-1)
2. 표면적으로는 유사 시스템 같지만, 주 기능이 아예 달라 삭제 [↑](#footnote-ref-2)
3. 10.3 태스크 일정 변경으로 인함 [↑](#footnote-ref-3)
4. 10.3 태스크 일정 변경으로 인함 [↑](#footnote-ref-4)